Operator	Description	Associativity
( )	Parentheses (function call) (see Note 1)	left-to-right
[ ]	Brackets (array subscript)	
.	Member selection via object name	
->	Member selection via pointer	
++ --	Postfix increment/decrement (see Note 2)	
++ --	Prefix increment/decrement	right-to-left
+ -	Unary plus/minus	
! ~	Logical negation/bitwise complement	
(*type*)	Cast (convert value to temporary value of *type*)	
*	Dereference	
&	Address (of operand)	
sizeof	Determine size in bytes on this implementation	
* / %	Multiplication/division/modulus	left-to-right
+ -	Addition/subtraction	left-to-right
<< >>	Bitwise shift left, Bitwise shift right	left-to-right
< <=	Relational less than/less than or equal to	left-to-right
> >=	Relational greater than/greater than or equal to	
== !=	Relational is equal to/is not equal to	left-to-right
&	Bitwise AND	left-to-right
^	Bitwise exclusive OR	left-to-right
|	Bitwise inclusive OR	left-to-right
&&	Logical AND	left-to-right
| |	Logical OR	left-to-right
? :	Ternary conditional	right-to-left
=	Assignment	right-to-left
+= -=	Addition/subtraction assignment	
*= /=	Multiplication/division assignment	
%= &=	Modulus/bitwise AND assignment	
^= |=	Bitwise exclusive/inclusive OR assignment	
<<= >>=	Bitwise shift left/right assignment	
,	Comma (separate expressions)	left-to-right

CSE101-Lec#2-First Part

- Operators

- In this lecture we will study
  - Operators
  - Types of Operators

Operators

- Operator is the symbol which performs some operations on the operands.

5+5=10

+ and = are the operator and
5 and 10 are operands

Types of Operators

- **Types of operators are:**
  1. Arithmetic operator
  2. Unary operator
  3. Relational operator
  4. Logical operator
  5. Assignment operator
  6. Conditional operator
  7. Bitwise operator
  8. Special operator

Description of Operators

- Arithmetic Operators

These are binary operators i.e. expression requires two operands

Operator	Description	Example (a=4 and b=2)
+	Addition of two operands	a + b = 6
-	Subtraction of two operands	a – b = 2
*	Multiplication of two operands	a * b = 8
/	Division of two operands	a / b = 2
%	Modulus gives the remainder after division of two operands	a % b = 0

Arithmetic Operators

If the radius of car wheel is 15inch then what will the diameter and calculate distance traveled after one rotation of that wheel?

Arithmetic Operators

Sol:

r = 15

diameter = r + r = 2 * r = 2 * 15 = 30

dist_travelled = pi * d

dist_travelled = pi * diameter

= 3.14 * 30 = 94.2

Arithmetic Operators

To get the remainder of the integer value.

Eg:

14 mod 3 = 2

17 mod 2 = 1

190 mod 3 = 1

```
3)14(4
   12
  ----
    2
```

Q:Suppose we have to distribute 10 chocolates among 3 students equally then after equal distribution how many chocolates will be left?

Sol: 10 mod 3 = 1

So 1 chocolate will be left as all 3 students will have 3 chocolates each.

Q1

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int i = -3;
    int k = i % 2;
    printf("%d\n", k);
    return 0;
}
```

A. Compile time error

B. -1

C. 1

D. None of these

Q2

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int i = 3;
    int l = i / -2;
    int k = i % -2;
    printf("%d %d\n", l, k);
    return 0;
}
```

A. Compile time error
B. -1 1
C. 1 -1
D. None of these

Q3

What will be the final value of x in the following C code?

```
#include <stdio.h>
int main()
{
    int x = 5 * 9 / 3 + 9;
    printf("%d",x);
    return 0;
}
```

A. 3.75

B. Depends on compiler

C. 24

D. 3

Q4

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int x = 5.3 % 2;
    printf("Value of x is %d", x);
    return 0;
}
```

A. Value of x is 2.3

B. Value of x is 1

C. Value of x is 0.3

D. Compile time error

Q5

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int a = 10;
    double b = 5.6;
    int c;
    c = a + b;
    printf("%d", c);
    return 0;
}
```

A. 15
B. 16
C. 15.6
D. 10

Unary Operator

These operator requires only one operand.

Operator	Description	Example(count=1)
+	unary plus is used to show positive value	+count; value is 1
-	unary minus negates the value of operand	-count; value is -1
++	Increment operator is used to increase the operand value by 1	++count; value is 2 count++; value is 2
--	Decrement operator is used to decrease the operand value by 1	--count; value is 1 count--; value is 1

++count increments count by 1 and then uses its value as the value of the expression. This is known a **prefix operator**.
count++ uses count as the value of the expression and then increments count by 1. This is known as **postfix operator**.

Difference between Prefix and Postfix

- Unary Prefix increment/ decrement performs the operation first, and then the value is assigned/ or used

Example:

Consider x=2, then

y = ++x; is equivalent to writing

//x = x + 1;

//y = x;

So eventually x will be incremented by 1, i.e x will become 3, and then the value 3 will be assigned to y

- Unary Postfix increment/ decrement will assign/ or use the value first and then the operation is performed

Example:

Consider x=2, then

y = x++; is equivalent to writing

//y = x;

//x=x+1;

Here y will take value 2, and then the value of x will be increment by 1, and x becomes 3.

Difference between Prefix and Postfix

- Example:

```
#include<stdio.h>
int main() {
  int x = 3, y, z;
  y = x++;
  z = ++x;
  printf("\n%d,%d,%d",x,y,z);
  return 0;
}
```

Output:

```
5, 3, 5
```

Explanation:

- Initialize x to 3
- Assign y the value we get by evaluating the expression x++, i.e, the value of x before increment then increment x.
- Increment x then assign z the value we get by evaluating the expression ++x, i.e, value of x after the increment.
- Print these values

Q1

What will be the output of following code?

```
#include <stdio.h>
    int main()
    {
        int a=1,b=1,c;
        c = a++ + b;
        printf("%d,%d,%d", a,b,c);
        return 0;
    }
```

A. 2,1,1

B. 1,2,1

C. 2,1,2

D. 1,1,2

Q2

What will be the output of following code?

```
#include <stdio.h>
    int main()
    {
        int d, a = 1, b = 2;
        d =  a++ + ++b;
        printf("%d %d %d", d, a, b);
        return 0;
    }
```

A. 4 2 2

B. 3 1 2

C. 4 2 3

D. 3 2 3

Q3

What will be the output of following code?

```
#include <stdio.h>
    int main()
    {
        int i = 0;
        int x = i++;
        int y = ++i;
        printf("%d % d\n", x, y);
        return 0;
    }
```

A. 0, 2
B. 0, 1
C. 1, 2
D. 1, 1

Q4

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int x = 4, y, z;
    y = --x;
    z = x--;
    printf("%d%d%d", x,  y, z);
    return 0;
}
```

A. 3 2 3
B. 2 3 3
C. 3 2 2
D. 2 3 4

Relational Operator

It compares two operands depending upon the their relation. Expression generates zero(false) or nonzero(true) value.

Operator	Description	Example (a=10 and b=20)
<	less than, checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(a < b) value is 1(true)
<=	less than or equal to, checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(a <= b) value is 1 (true).
>	greater than, checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(a > b) value is 0 (false).
>=	greater than or equal to, checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(a >= b) value is 0 (false).
==	equality ,checks if the value of two operands is equal or not, if yes then condition becomes true.	(a == b) value is 0 (false).
!=	inequality, checks if the value of two operands is equal or not, if values are not equal then condition becomes true.	(a != b) value is 1 (true).

Relational Operator

Q: Age of Sam is 20 and age of Tom is 19.

Verify the relationship between their age.

Sol: age of Sam = S1 = 20

age of Tom = T1 = 19

S1 < T1 = 0 (false)

S1 > T1 = 1 (true)

So, Sam is elder than Tom.

S1 == T1 = 0 (false)

Q1

What will be the output of following code?

```
#include <stdio.h>
    int main()
    {
        int a=1,b=2,c;
        c=a>b;
        printf("\n%d",c);
        return 0;
    }
```

A. 0

B. 1

C. 2

D. None of these

Q2

What will be the output of following code?

```
#include <stdio.h>
  int main()
  {
    int a=1,b=2;
    printf("\n%d",a!=b);
    return 0;
  }
```

A. 0

B. 1

C. 2

D. None of these

Q3

What will be the final value of d in the following C code?

```
#include <stdio.h>
int main()
{
    int a = 10, b = 5, c = 5;
    int d;
    d = b + c == a;
    printf("%d", d);
    return 0;
}
```

A. Syntax error
B. 1
C. 5
D. 10

Logical Operator

It checks the logical relationship between two expressions and the result is zero(false) or nonzero(true).

Operator	Description	Example
&&	Logical AND operator. If both the operands are true then condition becomes true.	(5>3 && 5<10) value is 1 (true).
| |	Logical OR Operator. If any of the two operands is true then condition becomes true.	(5>3 || 5<2) value is 1 (true).
!	Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(8==8) value is 0 (false).

Logical Operator

Grade system :

If (Marks >=90 || marks == 100)

    students performance is excellent.

If (Marks <= 40 && attendance < 75)

    student is detained.

Q1

//What will be the output of following code?

```
#include <stdio.h>
int main()
{
    int a = 10, b = 0,c;
    c=a&&b;
    printf("%d",c);
}
```

A. 0

B. 1

C. -1

D. None of these

Q2

What will be the output of following code?

```
#include <stdio.h>
int main()
{
    int a = 10, b = 0,c=2,d;
    d=a&&b||c-2;
    printf("%d",d);
}
```

A. 0
B. 1
C. -1
D. None of these

Q3

What will be the output of the following C code?

```
#include <stdio.h>
int main()
{
    int x = 1, y = 0, z = 5;
    int a = x && y || z++;
    printf("%d", z);
    return 0;
}
```

A. 6

B. 5

C. 0

D. None of these

Q4

What will be the output of following code?

```
#include <stdio.h>
int main()
{
    int x = 1, y = 0, z = 5;
    int a = x && y && z++;
    printf("%d", z);
    return 0;
}
```

A. 6
B. 5
C. 0
D. None of these